Amendments to the Claims

Please amend Claims 1, 5 and 6. The Claim Listing below will replace all prior versions of the claims in the application:

Claim Listing

1. (Currently amended) A Programmable Streaming Data Processor (PSDP), ~~which is~~ arranged to perform primitive initial processing functions directly on a set of data ~~received from a streaming data interface, PSDP performing initial processing on a set of data~~ comprising:

    a streaming data interface, ~~for receiving~~ arranged to receive data from a streaming data source;

    a streaming interface First In First Out (FIFO), arranged ~~for~~ to temporarily store ~~storing~~ streaming data from the streaming data interface;

    a data engine, arranged to receive output data from the streaming interface FIFO, ~~the data engine for determining~~ determine field boundaries therein, and ~~for processing~~ process fields to select one or more fields to be assembled into output tuples, the data engine also containing logic arranged to determine whether an output tuple is to be selected for further processing by additional Job Processing Units (JPUs)[[,]] and ~~for asserting~~ to assert a use or lose decision value according to that determination;

    a tuple generator, arranged to assemble ~~for assembling~~ fields into the output tuple[[,]] and, if the use or lose decision value indicates that such output tuple is to be discarded, ~~for preventing such~~ to prevent the output tuple ~~set~~ from being transferred for further processing by ~~from the output FIFO to the memory of~~ the JPU; and

    an output FIFO device, ~~for forming~~ arranged to temporarily store tuples ~~and temporarily storing them~~ prior to conditionally forwarding them to the JPU.

2. (Previously presented) An apparatus as in claim 1 wherein the use or lose decision value indicates a result from logic processing of fields read from the streaming data interface.

3.   (Previously presented) An apparatus as in claim 1 wherein the use or lose decision value indicates a result from Transaction Identifier (TID) processing.

4.   (Original) An apparatus as in claim 3 wherein the TID processing and data engine logic execute in parallel.

5.   (Currently amended) An apparatus as in claim 1 wherein the output tuple is greater in length than an expected predetermined size, and the ~~use/lose~~ use or lose decision value is then used to set an overflow field in the output tuple.

6.   (Currently amended) An apparatus as in claim 5 wherein the ~~use/lose~~ use or lose decision value is not asserted when a buffer local to the programmable data streaming processor is full; and

means for appending an overflow filter bit to a tuple that indicates a transfer of a tuple that should be ignored.

7.   (Previously presented) An apparatus as in claim 1 additionally comprising:

a Direct Memory Access (DMA) interface, coupled to the output FIFO, to provide direct access to a memory in the JPU.

8.   (Previously presented) An apparatus as in claim 1 wherein the use or lose decision value is used to reset the output FIFO write pointer so any prior fields in the present tuple are discarded.

9.   (Previously presented) An apparatus as in claim 1 wherein the overflow filter bit is inserted in a length field appended to record fragments.

10.   (Previously presented) An apparatus as in claim 1 wherein an invalid field is appended to a tuple to indicate the results of TID processing.

11. (Previously presented) An apparatus as in claim 10 wherein the invalid field indicates that the TID mode marks return tuple.

12. (Original) An apparatus as in claim 10 wherein the invalid field indicates that the tuple should not have been returned but the output FIFO overflowed.

13. (Previously presented) An apparatus as in claim 1 further comprising:
    a register reflecting the final PSDP status which is read by a Central Processing Unit (CPU) to identify whether any overflow or TID status bits are set in any of the tuples.

14. (Previously presented) An apparatus as in claim 1 wherein the use or lose decision value represents DeMorgan's Law reduction of multiple instructions.